应对后续多个 publishOn 操作可能产生的问题，并对最后一个元素之前（不包括最后一个元素）所重复的 publishOn 操作屏蔽绝大部分功能（具体前面已经介绍得很清楚了）。从 FluxGenerate.GenerateSubscription 到 PublishOnSubscriber，可以感受到 QueueSubscription 带给我们的穿针引线的效应，可以在多个类间进行状态控制。希望读者同样可以掌握：一个接口（QueueSubscription）、一个本地状态（类中的 sourceMode）、一个链式调用方法（onSubscribe，后者根据前者的结果来进行自己的行为设定）。

3.3 深入解读 subscribeOn

为了方便理解，首先通过图 3-2 来展示 subscribeOn 操作的行为细节。

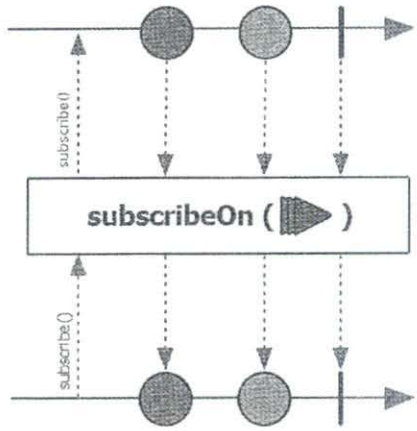

图 3-2

subscribeOn 操作主要针对的是发生订阅的线程，也就是对生产初始元素的线程的设定。注意，在图 3-2 中，在调用 subscribe 方法后会切换线程。

subscribeOn 操作的实现逻辑与 RxJava 2 中的 Flowable#subscribeOn(io.reactivex.Scheduler, boolean)的实现逻辑如出一辙，与 publishOn 操作相比，其相对简单。在本节中，我们就来探其究竟。

其实这里有一个比较有意思的“玩法”——请求异步化，这也是开发中容易忽视的东西。当使用 Flux#create(Consumer,FluxSink.OverflowStrategy)作为源的时候，我们可以在生产元素的过程中执行一些阻塞等待操作，也就是说，很可能造成元素的消费速度大于生产速度。如下面